

# CS190C Lec9

## Resource Calculation

# Overview

- Peak memory calculation
- FLOPs calculation
- What is the maximum batch size?
- What is the training time consumption?

## **PART1: Peak memory calculation**

For certain memory resource, how can we decide some scale of allocation?

- `batch_size`
- `max_seq_len`

We need to calculate peak memory, in order to ensure not suffering from `Cuda Out Of Memory` .

Suppose: `dtype=float32`, `d_ff = 4d_model`, `num_heads*d_qk = d_model`, `d_qk=d_v`, use `SwiGLU` for FFN, use `AdamW`.

We need to calculate in 4 parts:

- Parameters of model
- Activations of model during forward process
- Gradients
- Parameters of optimizer

# 1. Parameters of model

We need to calculate in 3 parts:

- Input layer
- Hidden layer
- Output layer

Let:  $b = \text{batch\_size}$ ,  $c = \text{seq\_len}$ ,  $d = \text{d\_model}$ ,  $v = \text{vocab\_size}$ ,  $h = \text{num\_heads}$ ,  $L = \text{num\_layers}$

# 1. Parameters of model

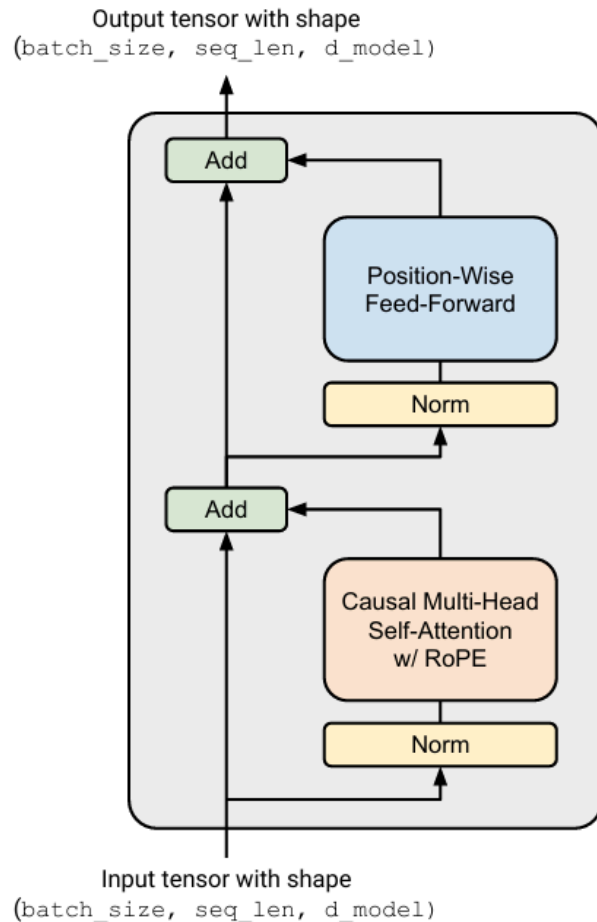
Input layer contains word embedding matrix only:

- `vocab_size*d_model` parameters in total.

# 1. Parameters of model

Hidden layer contains `num_layers` Transformer Blocks. For each block:

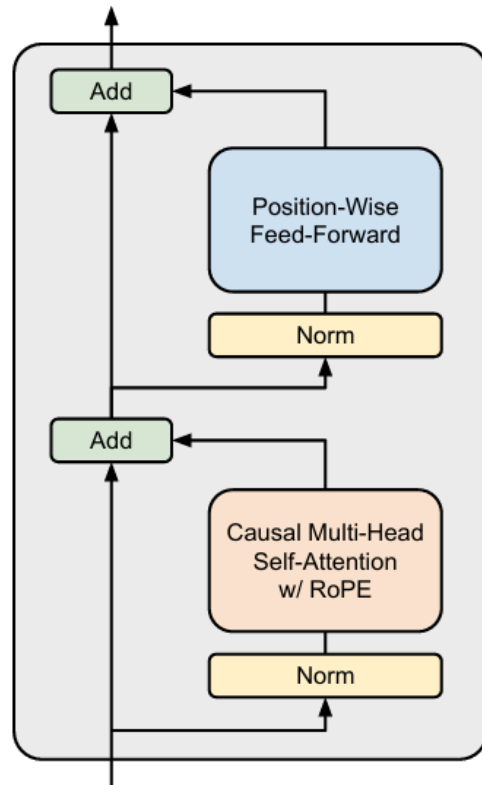
- 2 `RMSNorm` modules: each contains `d_model` parameters,  $2d$  in total.
- 1 `MHA` module:
  - $W_Q, W_K, W_V$ : shape = `d_model*d_model` ,  $3d^2$  in total.
  - $W_O$ : shape = `d_model*d_model` ,  $d^2$  in total.
  - $4d^2$  in total.



# 1. Parameters of model

Hidden layer contains `num_layers` Transformer Blocks. For each block:

Output tensor with shape  
(batch\_size, seq\_len, d\_model)



Input tensor with shape  
(batch\_size, seq\_len, d\_model)

- 1 FFN module:
  - $W_1$ :  $d_{\text{model}} * d_{\text{ff}} = 4 * d_{\text{model}} * d_{\text{model}}$
  - $W_3$ :  $d_{\text{model}} * d_{\text{ff}} = 4 * d_{\text{model}} * d_{\text{model}}$
  - $W_2$ :  $d_{\text{ff}} * d_{\text{model}} = 4 * d_{\text{model}} * d_{\text{model}}$
  - $12d^2$  in total
- $L(2d + 16d^2)$  in total

# 1. Parameters of model

Output layer contains Final `RMSNorm` and a `Linear Projection` module.

- Final `RMSNorm`: `d_model` parameters
- `Linear Projection`: `d_model*vocab_size` parameters
- $d + dv$  in total

# 1. Parameters of model

Total memory of this part:

- Input layer:  $\text{vocab\_size} * d_{\text{model}}$
- Hidden layer:  $\text{num\_layers} * (2 * d_{\text{model}} + 16 * d_{\text{model}} * d_{\text{model}})$
- Output layer:  $d_{\text{model}} + d_{\text{model}} * \text{vocab\_size}$
- Total parameter:  $N_p = 2dv + d + L(2d + 16d^2)$
- Total memory:  $4 * N_p$  bytes

## 2. Activations of model during forward process

We should consider following modules:

- Transformer Blocks
  - Activations of RMSNorm s
  - Activations of Multi-Head Attention
  - Activations of FFN
- Final RMSNorm
- Logits of Linear Projection (output embeddings)

## 2. Activations of model during forward process

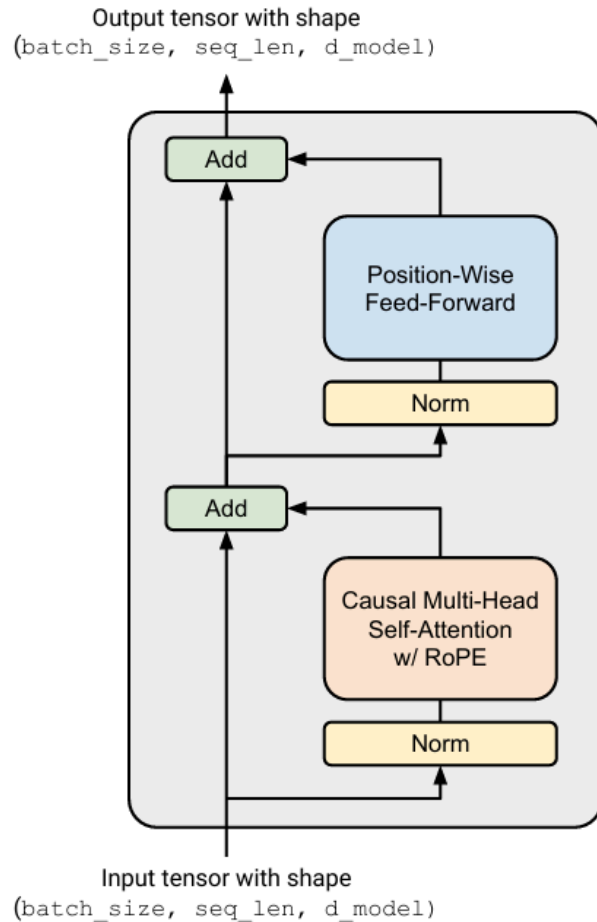
Why we need to save activations, instead of discard it right after calculated?

- We need to backward and calculate gradients after forward completed.
- Gradient calculation is related to activations!
- So we should infer what to save according to gradient calculation.

So how do we derivate what activation to save?

- Related to backward: Nodes on the computation graph.
- Related to parameter optimization.

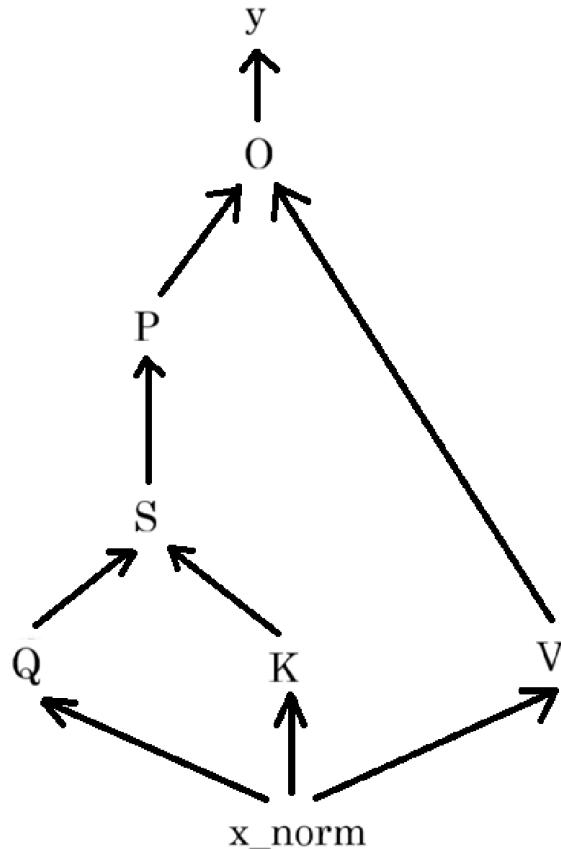
## 2. Activations of model during forward process



For RMSNorm before MHA :

- Forward:  $x_{norm} = \frac{x}{\sqrt{\text{RMS}(x)}} \cdot g$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial x_{norm}} \frac{\partial x_{norm}}{\partial x}$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial g} = \frac{\partial L}{\partial x_{norm}} \frac{\partial x_{norm}}{\partial g}$
- Activations to save:
  - $\frac{\partial x_{norm}}{\partial x} = f(x, g)$ , containing  $x$  ( $g$  has been considered in model parameters)
  - $\frac{\partial x_{norm}}{\partial g} = \frac{x}{\sqrt{\text{RMS}(x)}}$ , containing  $x$
  - Save  $x$ .
- Shape: [bsz, seq\_len, d\_model] ; Numbers:  $bcd$

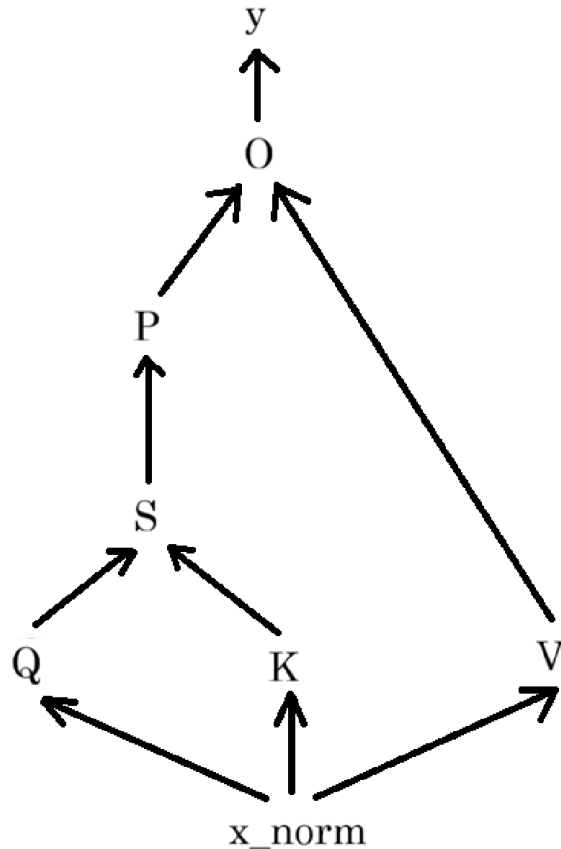
## 2. Activations of model during forward process



For calculation of  $Q$   $K$   $V$  in MHA :

- Forward:  $Q = W_Q x_{norm}$ ,  $K, V$  the same.
- Gradients calculation:
  - Gradients to calculate for backward:
$$\frac{\partial L}{\partial x_{norm}} = \frac{\partial L}{\partial Q} \frac{\partial Q}{\partial x_{norm}} + \frac{\partial L}{\partial K} \frac{\partial K}{\partial x_{norm}} + \frac{\partial L}{\partial V} \frac{\partial V}{\partial x_{norm}}$$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial W_Q} = \frac{\partial L}{\partial Q} \frac{\partial Q}{\partial W_Q}$ ,  $W_Q, W_K$  the same.
- Activations to save:
  - $\frac{\partial Q}{\partial x_{norm}} = W_Q$ , has been considered.
  - $\frac{\partial Q}{\partial W_Q} = x_{norm}$ .
  - Save  $x_{norm}$ .
- Shape: `[bsz, seq_len, d_model]` ; Numbers:  $bcd$

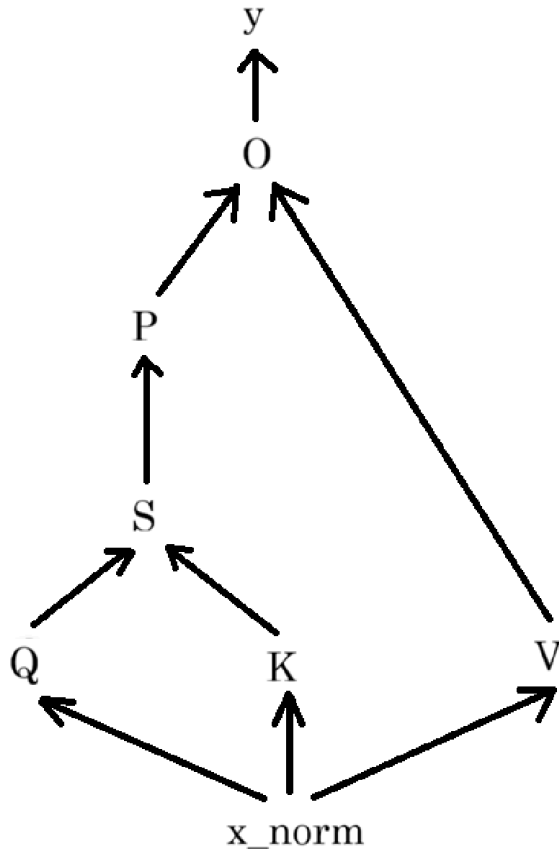
## 2. Activations of model during forward process



For calculation of Attention Score `S` in `MHA` :

- Forward:  $S = \frac{Q^T K}{\sqrt{d_k}}$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial Q} = \frac{\partial L}{\partial S} \frac{\partial S}{\partial Q}$ ,  $\frac{\partial L}{\partial K}$  is the same.
  - No learnable parameters.
- Activations to save:
  - $\frac{\partial S}{\partial Q} = K$ ,  $\frac{\partial S}{\partial K} = Q$
  - Save `Q`, `K`
- Shape: `[bsz, seq_len, d_model]` ; Numbers: `2bcd`

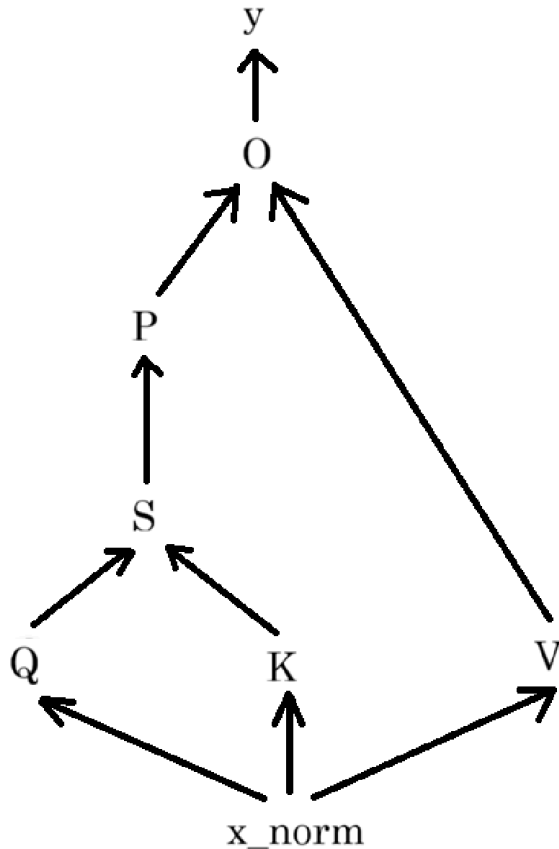
## 2. Activations of model during forward process



For calculation of Attention Weight `P` in `MHA` :

- Forward:  $P = \text{Softmax}(S)$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial S} = \frac{\partial L}{\partial P} \frac{\partial P}{\partial S}$
  - No learnable parameters.
- Activations to save:
  - $\frac{\partial P}{\partial S} = f(P)$
  - Save `P`
- Shape: `[bsz, num_heads, seq_len, seq_len]` ; Numbers:  $bhc^2$

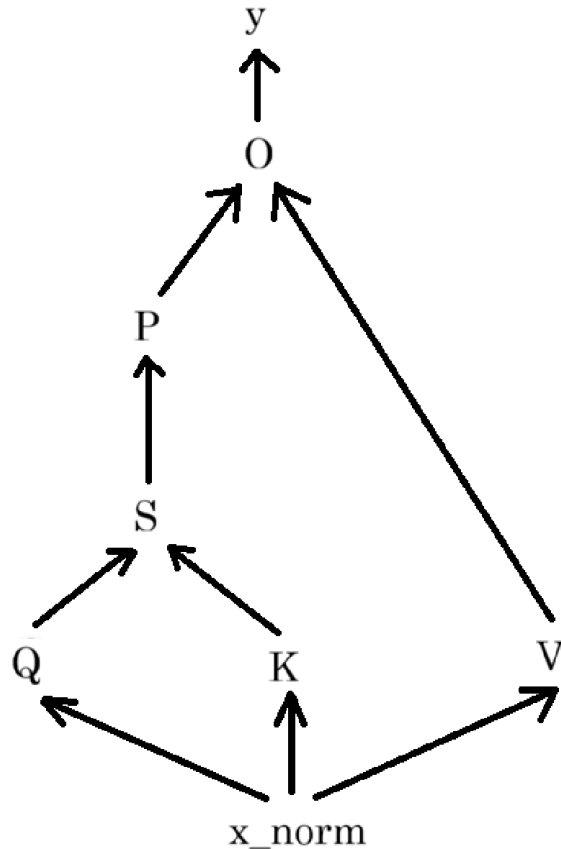
## 2. Activations of model during forward process



For calculation of multi-head output `O` in `MHA` :

- Forward:  $O = PV$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial P} = \frac{\partial L}{\partial O} \frac{\partial O}{\partial P}$ ,  $V$  is the same.
  - No learnable parameters.
- Activations to save:
  - $\frac{\partial O}{\partial P} = V$ ,  $\frac{\partial O}{\partial V} = P$
  - Save `V` ( `P` has been saved)
- Shape: `[bsz, seq, d_model]` ; Numbers: `bcd`

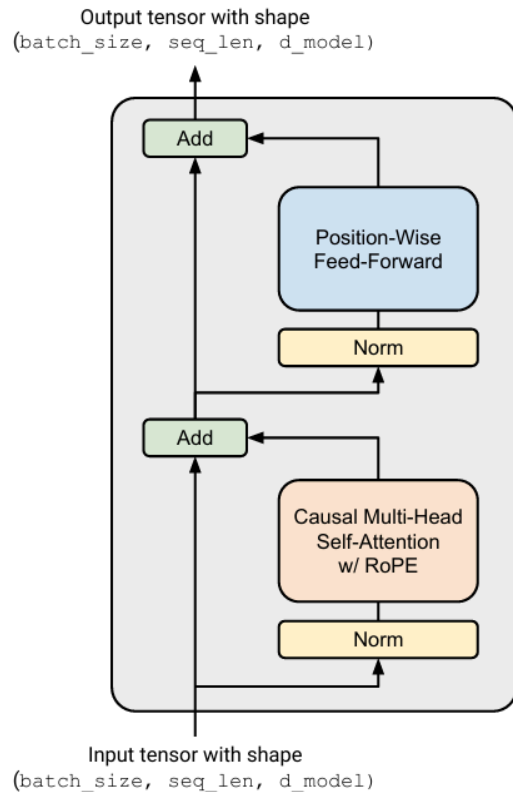
## 2. Activations of model during forward process



For calculation of attention output `y` in MHA :

- Forward:  $y = W_O O$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial O} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial O}$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial W_O} = \frac{\partial L}{\partial y} \frac{\partial y}{\partial W_O}$
- Activations to save:
  - $\frac{\partial y}{\partial O} = W_O$  (has been saved)
  - $\frac{\partial y}{\partial W_O} = O$
  - Save `O`
- Shape: `[bsz, seq, d_model]` ; Numbers: `bcd`

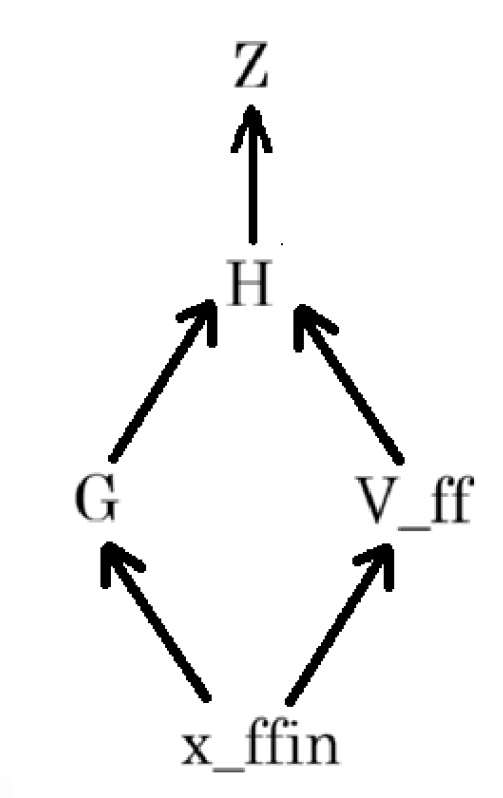
## 2. Activations of model during forward process



For residual connection and **RMSNorm** before **FFN** :

- Forward:  $x_{res} = x + y$ ,  $x_{ffin} = \frac{x_{res}}{\sqrt{\text{RMS}(x_{res})}} \cdot g_2$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial y} = \frac{\partial L}{\partial x_{ffin}} \frac{\partial x_{ffin}}{\partial x_{res}} \frac{\partial x_{res}}{\partial y}$
  - Gradients for  $g_2$
- Activations to save:
  - Save **x\_res**
- Shape: **[bsz, seq, d\_model]** ; Numbers: *bcd*

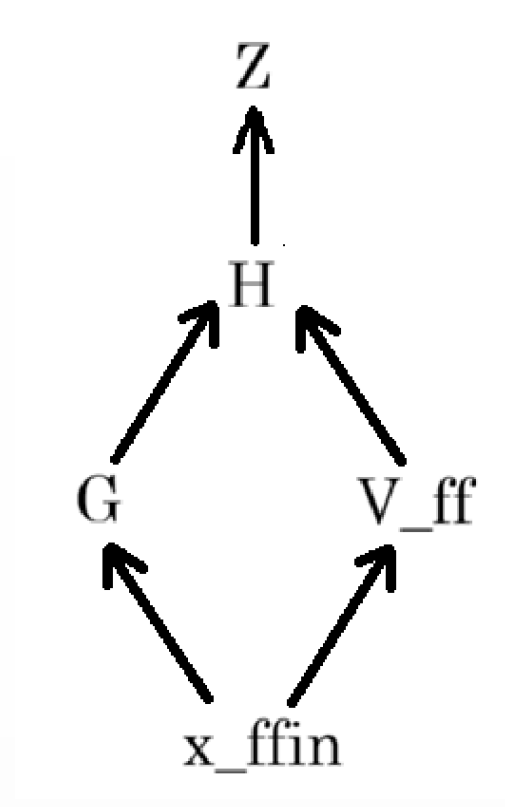
## 2. Activations of model during forward process



For W1 Gate in FFN :

- Forward:  $G = W_1 x_{ffin}$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial x_{ffin}} = \frac{\partial L}{\partial G} \frac{\partial G}{\partial x_{ffin}}$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial W_1} = \frac{\partial L}{\partial G} \frac{\partial G}{\partial W_1}$
- Activations to save:
  - $\frac{\partial G}{\partial x_{ffin}} = W_1$  (has been saved)
  - $\frac{\partial G}{\partial W_1} = x_{ffin}$
  - Save `x_ffin`
- Shape: `[bsz, seq, d_model]` ; Numbers: *bcd*

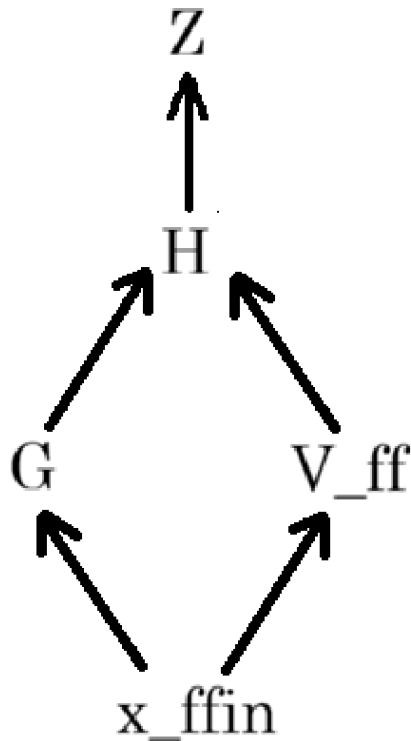
## 2. Activations of model during forward process



For W3 Gate in FFN :

- Forward:  $V_{ff} = W_3 x_{ffin}$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial x_{ffin}} = \frac{\partial L}{\partial V_{ff}} \frac{\partial V_{ff}}{\partial x_{ffin}}$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial W_3} = \frac{\partial L}{\partial V_{ff}} \frac{\partial V_{ff}}{\partial W_3}$
- Activations to save:
  - $\frac{\partial V_{ff}}{\partial x_{ffin}} = W_3$  (has been saved)
  - $\frac{\partial V_{ff}}{\partial W_3} = x_{ffin}$  (has been saved)
  - Do not need to save anything.

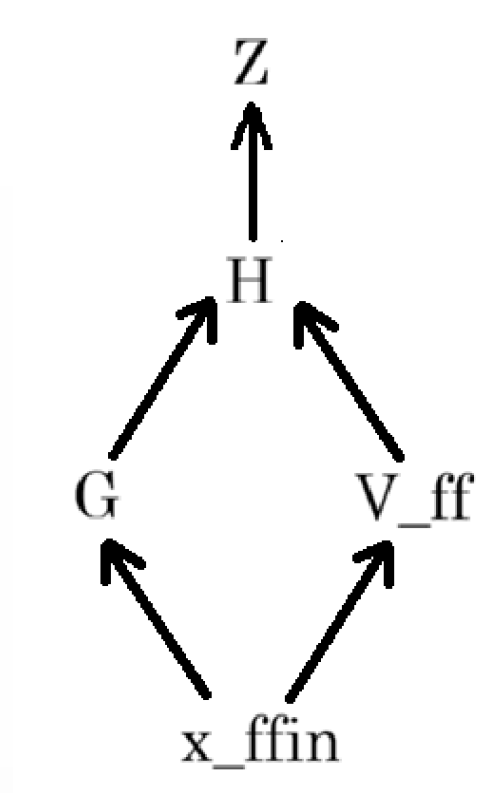
## 2. Activations of model during forward process



For gate interaction in FFN :

- Forward:  $H = \text{SiLU}(G) \cdot V_{ff} = G \cdot \sigma(G) \cdot V_{ff}$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial V_{ff}} = \frac{\partial L}{\partial H} \frac{\partial H}{\partial V_{ff}}$ ,  
 $\frac{\partial L}{\partial G} = \frac{\partial L}{\partial H} \frac{\partial H}{\partial G}$
  - No learnable parameters.
- Activations to save:
  - $\frac{\partial H}{\partial V_{ff}} = \text{SiLU}(G)$
  - $\frac{\partial H}{\partial G} = f(G, V_{ff})$
  - Save **G** , **V\_ff**
- Shape: **[bsz, seq, d\_ff]** ; Numbers:  $8bcd$

## 2. Activations of model during forward process

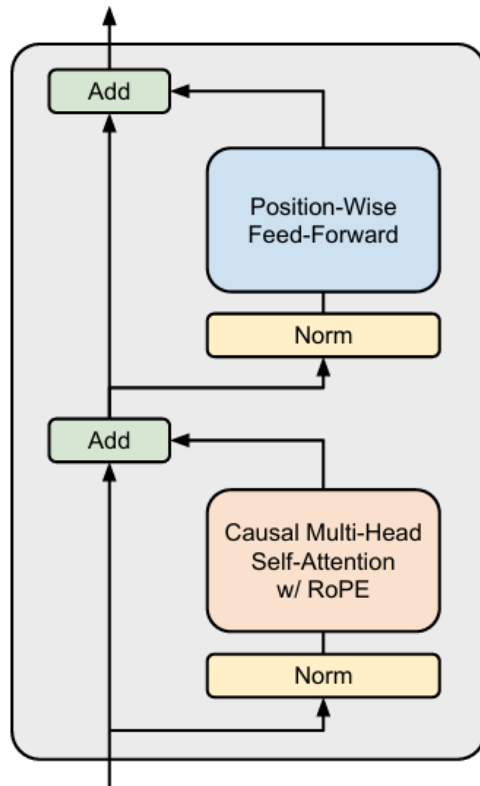


For W2 Gate in FFN :

- Forward:  $Z = W_2 H$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial H} = \frac{\partial L}{\partial Z} \frac{\partial Z}{\partial H}$
  - Gradients to calculate for optimization:  $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial Z} \frac{\partial Z}{\partial W_2}$
- Activations to save:
  - $\frac{\partial Z}{\partial H} = W_2$  (has been saved)
  - $\frac{\partial Z}{\partial W_2} = H$
  - Save H
- Shape: `[bsz, seq, d_ff]` ; Numbers: `4bcd`

## 2. Activations of model during forward process

Output tensor with shape  
(batch\_size, seq\_len, d\_model)

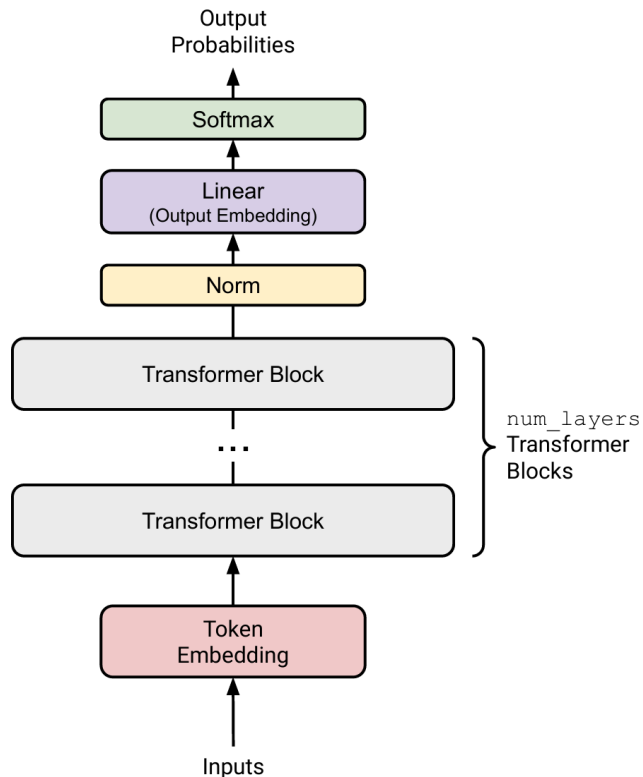


Input tensor with shape  
(batch\_size, seq\_len, d\_model)

For final residual:

- Forward:  $x_{out} = x_{res} + Z$
- Gradient portion=1, no learnable parameters
- No need to save anything.

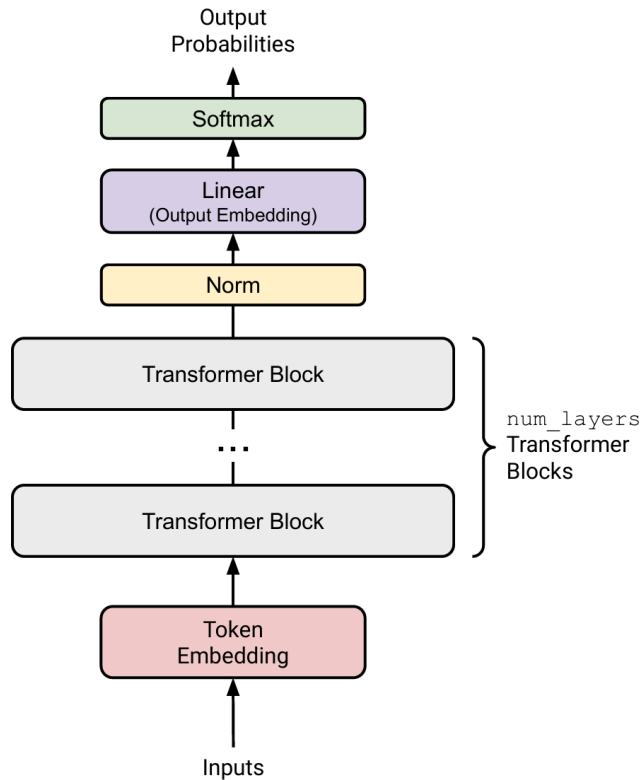
## 2. Activations of model during forward process



For Final RMSNorm :

- Forward:  $X_{final} = \frac{x_{final}}{\sqrt{\text{RMS}(x_{final})}} \cdot g_{final}$
- Similarly, save the input of RMSNorm , that is the output of the last layer of Transformer.
- Shape: `[bsz, seq, d_model]` ; Numbers: *bcd*

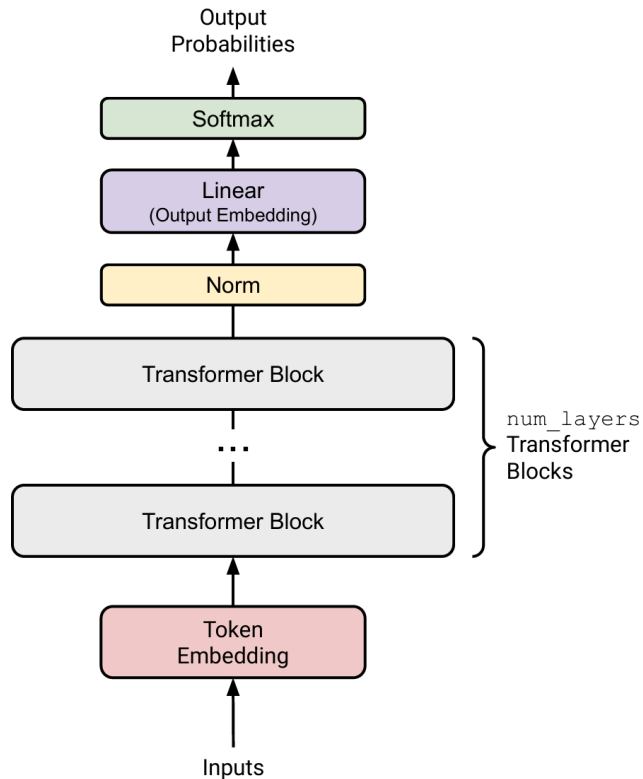
## 2. Activations of model during forward process



For **Output Embedding**:

- Forward:  $\text{Logits} = W_{vocab} X_{final}$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial X_{final}} = \frac{\partial L}{\partial \text{Logits}} \frac{\partial \text{Logits}}{\partial X_{final}}$
  - Gradients to calculate for optimization:
$$\frac{\partial L}{\partial W_{vocab}} = \frac{\partial L}{\partial \text{Logits}} \frac{\partial \text{Logits}}{\partial W_{vocab}}$$
- Activations to save:
  - $\frac{\partial \text{Logits}}{\partial X_{final}} = W_{vocab}$  (has been saved)
  - $\frac{\partial \text{Logits}}{\partial W_{vocab}} = X_{final}$
  - Save **X\_final**
- Shape: **[bsz, seq, d\_model]** ; Numbers: *bcd*

## 2. Activations of model during forward process



For **Output Embedding** :

- Forward:  $L = CE(\text{Logits}, \text{Standard})$
- Gradients calculation:
  - Gradients to calculate for backward:  $\frac{\partial L}{\partial \text{Logits}}$
  - No learnable parameters.
- Activations to save:
  - $\frac{\partial L}{\partial \text{Logits}} = f(\text{Logits})$
  - Save **Logits**
- Shape: **[bsz, seq, vocab\_size]** ; Numbers: *bcv*

## 2. Activations of model during forward process

- For each Transformer Block:  $20bcd + bhc^2 = N_B$  in total.
- For the whole transformer:  $L * N_B + 2bcd + bcv = N_A$  in total.
- $4 * N_A$  bytes in total.

### 3. Gradients

It's easy: The number of elements of gradients is the same as model parameter's elements.

So  $4 * N_p$  bytes in total

## 4. Parameters of optimizer

Parameters of AdamW:

- $m_t$ : historical gradient  $\Rightarrow$  the shape is the same as model gradients.
- $V_t$ : historical gradient fluctuation  $\Rightarrow$  the shape is the same as model gradients.

So  $2 * 4 * N_p = 8 * N_p$  in total.

## Peak memory calculation

The peak memory is:  $16 * N_p + 4 * N_A$

$$N_p = 2dv + d + L(2d + 16d^2)$$

$$N_A = L * N_B + 2bcd + bcv$$

$$N_B = 20bcd + bhc^2$$

So the peak memory is:

$$\begin{aligned} &16[2dv + d + L(2d + 16d^2)] + 4[L(20bcd + bhc^2) + 2bcd + bcv] \\ &= L(256d^2 + 32d + 80bcd + 4bhc^2) + (32dv + 16d + 8bcd + 4bcv) \end{aligned}$$

## PART2: FLOPs calculation

FLOPs: Floating Point Operations, which is used to measure how many computing resources will be consumed.

For example:  $y = AB$ ,  $A \in R^{n \times m}$ ,  $B \in R^{m \times k}$

- For each element  $y_{ij}$ :  $m$  multiplies &  $m - 1$  adds
- $m(nk)$  multiplies and  $(m - 1)(nk)$  adds, approximately  $2mnk$  FLOPs.

How many FLOPs for a single training iteration?

- FLOPs for a forward pass
- FLOPs for a backward pass
- FLOPs for an optimizer update pass

We mainly focus on **matrix calculation**, and neglect element-wise computation of RMSNorm , Softmax and so on.

# 1. FLOPs for a forward pass

For MHA module:

- Calculation of  $Q$   $K$   $V$  :
  - $W_{QKV}: [d, d]$   $x: [b, c, d]$
  - FLOPs:  $3b(2cdd) = 6bcd^2$
- Calculation of  $QK^T$ :
  - $Q: [b, h, c, d/h]$   $K^T: [b, h, d/h, c]$
  - FLOPs:  $bh(2ccd/h) = 2bc^2d$

# 1. FLOPs for a forward pass

For MHA module:

- Calculation of  $\text{Softmax} \cdot V = \text{Attn} \cdot V$ 
  - $\text{Attn}: [b, h, c, c]$   $V: [b, h, c, d/h]$
  - FLOPs:  $bh(2ccd/h) = 2bc^2d$
- Calculation of  $O$ :
  - $\text{Attn} \cdot V: [b, c, d]$   $W_O: [d, d]$
  - FLOPs:  $2bcd^2$
- In total:  $8bcd^2 + 4bc^2d$

# 1. FLOPs for a forward pass

For FFN module:

- $w_1$  and  $w_3$  projections:
  - input:  $[b, c, d]$  proj:  $[b, c, 4d]$  , 2 projections
  - FLOPs:  $2(8bcd^2) = 16bcd^2$
- $w_2$  projection:
  - input:  $[b, c, 4d]$  proj:  $[b, c, d]$
  - FLOPs:  $8bcd^2$
- In total:  $24bcd^2$

# 1. FLOPs for a forward pass

For a single Transformer Block:

- $32bcd^2 + 4bc^2d$  FLOPs in total
- $L(32bcd^2 + 4bc^2d)$  FLOPs for all blocks.

# 1. FLOPs for a forward pass

For output layer:

- input:[b,c,d] proj:[d,v]
- FLOPs:  $2bcdv$

So  $F_{forward} = L(32bcd^2 + 4bc^2d) + 2bcdv$  in total.

## 2. FLOPs for a backward pass

We only need to estimate it roughly.

For matrix calculation  $Y = XW$ ,  $X$  is the activation, a part of the computation graph,  $W$  is the learnable parameter.

For backward:

- We need to calculate  $\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \frac{\partial Y}{\partial X}$  (discussed in part 1.2), that is to calculate  $\frac{\partial L}{\partial Y} W^T$
- We also need to calculate  $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y} \frac{\partial Y}{\partial W}$ , that is to calculate  $\frac{\partial L}{\partial Y} X$

The sum of FLOPs of them is roughly equals to  $2F_{forward}$  (1 matrix calculation is correspond to 2)

### 3. FLOPs for an optimizer update pass

For AdamW:

- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
  - For each element: 2 multiplications, 1 addition.
  - 3 FLOPs for each element.
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
  - For each element: 3 multiplications, 1 addition.
  - 4 FLOPs for each element.
- $\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$ 
  - For each element:  $\sqrt{v_t}$ ,  $+\epsilon$ ,  $\frac{m_t}{\sqrt{v_t} + \epsilon}$ ,  $\alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$ ,  $\theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon}$
  - 5 FLOPs for each element

### 3. FLOPs for an optimizer update pass

For AdamW:

- $\theta_t = \theta_t - \alpha \lambda \theta_{t-1}$ 
  - For each element: 2 multiplications, 1 addition.
  - 3 FLOPs for each element
- In total:  $3 + 4 + 5 + 3 = 15$  FLOPs for each element
- Number of element is the number of model parameters
$$N_p = 2dv + d + L(2d + 16d^2)$$
- $F_{optimize} = 15N_p = 30dv + 15d + 15L(2d + 16d^2)$

## FLOPs calculation

FLOPs for a complete training iteration:

$$\begin{aligned} F_{total} &= F_{forward} + F_{backward} + F_{optimize} \\ &= 3F_{forward} + F_{optimize} \\ &= 3L(32bcd^2 + 4bc^2d) + 6bcdv + 30dv + 15d + 15L(2d + 16d^2) \end{aligned}$$

**PART3: What is the maximum batch size?**

Suppose we have only 1 rtx3090. What the maximum batch size we can allocate without gradient accumulation?

$$M_{peak} = [L(256d^2 + 32d) + (32dv + 16d)] + b[L(80cd + 4hc^2) + (8cd + 4cv)]$$

$$= M_{static} + bM_{per\_sample}$$

$$M_{3090} = 24GB = 24 * 1024^3$$

$$\Rightarrow b_{max} \approx \left\lfloor \frac{M_{3090} - M_{static}}{M_{per\_sample}} \right\rfloor$$

$$\Rightarrow b_{max} \approx \frac{24 \times 10^9 - (256Ld^2 + 32dv)}{4Lc(20d + hc)}$$

## **PART4: What is the training time consumption?**

Suppose we have 4 rtx3090, using `wikitext103` dataset to train 5 epochs. What the time consumption, neglecting time of evaluation?

Can we calculate like this?

$$T_{seconds} = \frac{\text{Total FLOPs}}{\text{Peak efficiency of GPU}}$$

We should consider `Model FLOPs Utilization` (MFU) of GPU. That is: GPUs are expected to work at efficiency  $\text{MFU} * \text{Peak}$

$$T_{seconds} = \frac{\text{Total FLOPs}}{\text{MFU} * \text{Peak}}$$

$$\text{Total FLOPs} = \text{Total Steps} * \text{FLOPs per Step}$$

$$\text{Total Steps} * bc = \text{Total Tokens} * \text{epoch}$$

$$\Rightarrow \text{Total Steps} = \frac{5 * \text{Total Tokens}}{bc} = \frac{5.15 * 10^8}{bc}$$

$$\text{Total FLOPs} = \frac{5.15 * 10^8}{bc} (3L(32bcd^2 + 4bc^2d) + 6bcdv + 30dv + 15d + 15L(2d + 16d^2))$$

$$T_{seconds} = \frac{5.15 * 10^8}{1.424 * 10^{14}} \frac{3L(32bcd^2 + 4bc^2d) + 6bcdv + 30dv + 15d + 15L(2d + 16d^2)}{bc * \text{MFU}}$$

$$T_{seconds} = 3.617 * 10^{-6} \cdot \frac{3L(32bcd^2 + 4bc^2d) + 6bcdv + 30dv + 15d + 15L(2d + 16d^2)}{bc * \text{MFU}}$$

Suppose  $b = 4, c = 256, d = 1024, L = 12, v = 32000, \text{MFU} = 0.5$

$$T_{seconds} = 3.617 * 10^{-6} \cdot \frac{3L(32bcd^2 + 4bc^2d) + 6bcdv + 30dv + 15d + 15L(2d + 16d^2)}{bc * \text{MFU}}$$

$$\Rightarrow T_{seconds} = 10461.9974$$

$$\Rightarrow T_{hours} = 2.9061$$